OVERVIEW OF THE NIST SECURE SOFTWARE DEVELOPMENT FRAMEWORK (SSDF)

KEVIN STINE, NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST)

KAREN SCARFONE, SCARFONE CYBERSECURITY

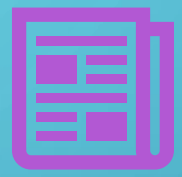
SSDF PUBLICATION INFORMATION

- NIST Cybersecurity White Paper, *Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)*
- Draft posted for public comment in June 2019
- Finalized in April 2020
- https://doi.org/10.6028/NIST.CSWP.04232020
- For both *software producers* (e.g., COTS vendors, government software developers, custom software developers) and *software consumers* (federal government agencies and other organizations)

APPROACH SIMILAR TO NIST CYBERSECURITY FRAMEWORK

Provides a common language to describe fundamental, sound secure software development practices

Can help an organization document its secure software development practices today and define its future target practices as part of its continuous improvement process



Leverages existing secure software development practices from established standards, guidance, and secure software development practice documents

Do no harm (to organizations who have already adopted established practices)

SSDF SCOPE

- Can be used by organizations in any sector or community, regardless of size or cybersecurity sophistication
- Can be applied to software developed to support IT, ICS, IoT, or cyber-physical systems (CPS)
- Can be integrated into any existing software development workflow and automated toolchain
- Broadly applicable—not specific to technologies, platforms, programming languages, SDLC models, development environments, operating environments, tools, etc.
- Can assist an organization in transitioning its secure software development practices for use with a modern software development model (e.g., agile, DevOps)

GOALS FOR SSDF PRACTICES

Flexible: Not all organizations have the same security objectives and priorities.

Customizable: Each software producer may have unique security assumptions, and each software consumer may have unique security needs.

Selective: Focus on the practices that will be the most helpful.

SSDF PRACTICE GROUPS

Prepare the Organization (PO): Ensure the organization's people, processes, and technology are prepared to perform secure software development at the organization level and, in some cases, also for each individual project.

Protect the Software (PS): Protect all components of the software from tampering and unauthorized access.

Produce Well-Secured Software (PW): Produce well-secured software that has minimal security vulnerabilities in its releases.

Respond to Vulnerabilities (RV): Identify vulnerabilities in software releases and respond appropriately to address those vulnerabilities and prevent similar vulnerabilities from occurring in the future.

ELEMENTS OF AN SSDF PRACTICE

Practices	Tasks	Implementation Examples	References
Configure the Software to Have Secure Settings by Default (PW.9): Help improve the security of the software at the time of installation to reduce the likelihood of the software being deployed with weak security settings that would put it at greater risk of compromise.	**PW.9.1:** Determine how to configure each setting that has an effect on security so that the default settings are secure and do not weaken the security functions provided by the platform, network infrastructure, or services.	• Conduct testing to ensure that the settings, including the default settings, are working as expected and are not inadvertently causing any security weaknesses, operational issues, or other problems.	**BSA**: CF.1, TC.1 **IDASOAR**: Fact Sheet 23 **ISO27034**: 7.3.5 **OWASPTEST**: Phase 4.2 **SCAGILE**: Tasks Requiring the Help of Security Experts 12 **SCSIC**: Vendor Software Delivery Integrity Controls, Vendor Software Development Integrity Controls **SP800181**: SP-DEV-002; K0009, K0039, K0073, K0153, K0165, K0275, K0531; S0167

Task: An individual action (or actions) needed to accomplish a practice

Implementation Example: An example of a type of tool, process, or other method that could be used to implement this practice

Reference: An established secure development practice document and its mappings to a particular task

REFERENCES

- **BSA**, Framework for Secure Software
- **Building Security In Maturity Model (BSIMM)** Version 10
- **Institute for Defense Analyses (IDA)**, State-of-the-Art Resources (SOAR) for Software Vulnerability Detection, Test, and Evaluation 2016
- **ISO/IEC**, Information technology – Security techniques – Application security – Part 1: Overview and concepts, ISO/IEC 27034-1:2011
- **Microsoft**, Security Development Lifecycle
- **NIST**:
  - Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1
  - Security and Privacy Controls for Federal Information Systems and Organizations, NIST Special Publication (SP) 800-53 Revision 4
  - Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems, NIST SP 800-160 Volume 1
  - National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework, NIST SP 800-181

REFERENCES (CONT.)

- **Open Web Application Security Project (OWASP):**
  - OWASP Application Security Verification Standard 4.0
  - OWASP Testing Guide 4.0
  - Software Assurance Maturity Model Version 1.5
- **Payment Card Industry (PCI) Security Standards Council,** Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures Version 1.0
- **Software Assurance Forum for Excellence in Code (SAFECode):**
  - Fundamental Practices for Secure Software Development: Essential Elements of a Secure Development Lifecycle Program, Third Edition
  - Managing Security Risks Inherent in the Use of Third-Party Components
  - Practical Security Stories and Security Tasks for Agile Development Environments
  - Software Integrity Controls: An Assurance-Based Approach to Minimizing Risks in the Software Supply Chain
  - Tactical Threat Modeling